DRAF: A Low-Power DRAM-Based Reconfigurable Acceleration Fabric

The DRAM-Based Reconfigurable Acceleration Fabric (DRAF) uses commodity DRAM technology to implement a bit-level, reconfigurable fabric that improves area density by 10 times and power consumption by more than 3 times over conventional field-programmable gate arrays. Latency overlapping and multicontext support allow DRAF to meet the performance and density requirements of demanding applications in datacenter and mobile environments.

Mingyu Gao
Stanford University

Christina Delimitrou
Cornell University

Dimin Niu
Krishna T. Malladi
Hongzhong Zheng
Bob Brennan
Samsung Semiconductor

Christos Kozyrakis
Stanford University

The end of Dennard scaling has made it imperative to turn toward application- and domain-specific acceleration as an energy-efficient way to improve performance.[1] Field-programmable gate arrays (FPGAs) have become a prominent acceleration platform as they achieve a good balance between flexibility and efficiency.[2] FPGAs have enabled accelerator designs for numerous domains, including datacenter computing,[3] in which applications are much more complex and change frequently, and multitenancy sharing is a principal way to achieve resource efficiency.

For FPGA-based accelerators to become widely adopted, their cost must remain low. This is an issue both for large-scale datacenters that are optimized for total cost of ownership and for small mobile devices that have strict budgets for power and chip area. Unfortunately, conventional FPGAs realize arbitrary bit-level logic functions using static RAM (SRAM) based lookup tables and configurable interconnects, both of which incur significant area and power overheads. The poor logic density and high power consumption limit the functionality that one can implement within an FPGA. Previous research has used networks of medium-sized FPGAs[3] or developed multicontext FPGAs[4] to circumvent these limitations, but these approaches come with their own overheads. For details, see the sidebar, "FPGAs in Datacenters and Multicontext Reconfigurable Fabrics."

We developed the DRAM-Based Reconfigurable Acceleration Fabric (DRAF), a reconfigurable fabric that improves logic density and reduces power consumption through the use of dense, commodity DRAM arrays. DRAF is bit-level reconfigurable and has similar flexibility as conventional FPGAs. DRAF

Published by the IEEE Computer Society

FPGAs in Datacenters and Multicontext Reconfigurable Fabrics

The advantages of spatial programmability and post-fabrication reconfigurability have made field-programmable gate arrays (FPGAs) the most successful and widely used reconfigurable fabric for accelerator designs in various domains. FPGAs provide bit-level reconfigurability through lookup tables, which can implement arbitrary combinational logic functions by storing the function outputs in small static RAM arrays. The typical lookup table granularity at the moment is 6-bit input and 1-bit output. FPGAs also have flip-flops for data retiming and temporary storage. The lookup tables and flip-flops are grouped into configurable logic blocks, which are organized into a 2D grid layout with other dedicated DSP and block RAM blocks. A bit-level, statically configurable interconnect supports communication between these blocks.

FPGAs have recently been used in datacenters as an acceleration fabric for cloud applications.[1–3] Datacenter servers often host multiple complex applications. Hence, multiple large FPGA devices are often necessary to provide sufficient resources for multiple large accelerators. Unfortunately, the tight power budget and the focus on total cost of ownership make it impractical to introduce expensive, power-hungry devices. To counteract these issues, Microsoft proposed the Catapult design, using medium-sized FPGAs with custom-designed interconnects linked between them.[1] Although it improves performance, this approach increases the system complexity and design integration cost, while still supporting only a single application on the acceleration fabric.

Multicontext reconfigurable fabrics[4] can support multitenancy sharing by allowing rapid runtime switch between multiple designs (contexts) that are all mapped onto a single fabric, similar to hardware-supported thread switching in multithreaded processors. Such fabrics store all context configurations on chip, either in specialized lookup tables[5] or in separate global backup memories.[6] Both approaches consume significant on-chip area for the additional storage, greatly reducing the single-context logic capacity. In addition, loading the configuration from the backup memories can result in long context switch latency. Because of their large overheads, multicontext FPGAs have not been widely adopted by industry.

References

1. A. Putnam et al., "A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services," *Proc. 41st Ann. Int'l Symp. Computer Architecture* (ISCA), 2014, pp. 13–24.
2. J. Hauswald et al., "Sirius: An Open End-to-End Voice and Vision Personal Assistant and Its Implications for Future Warehouse Scale Computers," *Proc. 20th Int'l Conf. Architectural Support for Programing Languages and Operating Systems* (ASPLOS), 2015, pp. 223–238.
3. R. Polig et al., "Giving Text Analytics a Boost," *IEEE Micro*, vol. 34, no. 4, 2014, pp. 6–14.
4. T.R. Halfhill, "Tabula's Time Machine," *Microprocessor Report*, vol. 131, 2010.
5. E. Tau et al., "A First Generation DPGA Implementation," *Proc. 3rd Canadian Workshop Field-Programmable Devices* (FPD), 1995, pp. 138–143.
6. S. Trimberger et al., "A Time-Multiplexed FPGA," *Proc. 5th IEEE Symp. FPGA-Based Custom Computing Machines* (FCCM), 1997, p. 22.

includes architectural optimizations, such as latency overlapping and multicontext support with fast context switching, that allow it to transform slow DRAM into a performant reconfigurable fabric suitable for both datacenters and mobile platforms.

Challenges for DRAM-Based FPGAs

Dense DRAM technology provides a new approach to realize high-density, low-power reconfigurable fabrics necessary in constrained environments such as datacenters and mobile devices. However, simply replacing the SRAM-based lookup tables in FPGAs with DRAM-based cell arrays would lead to critical challenges in logic utilization, performance, and even operation correctness. First, DRAM arrays are heavily optimized for area and cost efficiency by using very wide inputs (address) and outputs (data). Such wide granularity does not match the relatively fine-grained logic functions in most real-world accelerator designs, resulting in underutilization of the DRAM-based lookup tables. Simply reducing the I/O width of DRAM arrays would forfeit the density benefit, because the peripheral logic would now dominate the lookup table area. Second, DRAM access speed is 30 times slower than that of SRAM arrays (2 to 10 ns versus 0.1 to 0.5 ns). Without careful optimization, a 30 times slower FPGA would hardly provide any acceleration over programmable processors. Third, implementing large and complex logic functions often requires multiple lookup tables to be chained together, which is problematic with DRAM lookup tables, because

Table 1. Comparison of the DRAM-Based Reconfigurable Acceleration Fabric (DRAF) and a conventional field-programmable gate array (FPGA).

Features	Conventional FPGA	DRAF
Lookup table technology	Static RAM (SRAM)	DRAM
Lookup table delay	Short (0.1 to 1 ns)	Long (1 to 10 ns)
Lookup table output width	Single bit	Multiple bits
Logic capacity	Moderate	Very high
No. of configurations	Single	Multiple (4 to 8)
Power consumption	Moderate	Low

the destructive nature of DRAM accesses requires explicit activation and precharge operations with precise timing. Without careful management and coordination between lookup tables, the lookup table contents would be destroyed if accessed with an unstable input. Finally, DRAM requires periodic refresh operations, which could negatively impact system power consumption and application performance.

DRAF Architecture

DRAF leverages DRAM technology to implement a reconfigurable fabric with higher logic capacity and lower power consumption than conventional FPGAs. Table 1 summarizes the key features of DRAF as compared to a conventional FPGA.

DRAF implements several key architectural optimizations to overcome the challenges discussed in the previous section. First, it uses a specialized DRAM lookup table design that achieves both high density and high utilization by using a narrower output width and flexible column logic. Second, it uses a simple phase-based solution to specify the correct timing for each lookup table, and a three-way delay overlapping technique to significantly reduce the impact of DRAM operation latencies. Third, DRAF coordinates DRAM refresh in the device driver to reduce its power and latency impact. Finally, DRAF provides efficient multicontext support, which opens up the opportunity for sharing the acceleration fabric between multiple applications, greatly decreasing the overall system cost.

Overview

Similar to an FPGA, DRAF uses three types of logic blocks. The configurable logic block (CLB) contains lookup tables made with DRAM cell arrays and conventional flip-flops. The lookup table supports multiple on-chip configurations, each stored in one of the contexts. The digital signal processing (DSP) block is used for complex arithmetic operations, and the block RAM (BRAM) is for on-chip data storage. They are similar to those in FPGAs, but implemented in DRAM technology, which makes their latency and area worse than the corresponding implementation in a logic process. However, as we will show, the DRAM-based lookup table will also have much higher latency than an SRAM-based lookup table; therefore, the increased latencies of DSP and BRAM are not critical and do not dominate the overall design critical path. In addition, the combinational DSP logic does not need to be replicated across contexts, thus offsetting its area overhead. The DRAM array in the BRAM block is similar to the lookup tables, but with larger capacity, and is used for data storage rather than design configurations.

The blocks in DRAF are organized in a 2D grid layout similar to that of conventional FPGAs (see Figure 1a). The DRAF interconnect uses a simple and static time-multiplexing scheme to support multiple contexts.[5]

Configurable Logic Block

Figure 1b shows the structure of the CLB in DRAF. The density advantage of DRAM technology allows a DRAF CLB to provide 10 times the logic capacity over an FPGA CLB within the same area. The CLB contains a few DRAM-based lookup tables and the associated flip-flops and auxiliary multiplexers. The inputs of the lookup table are

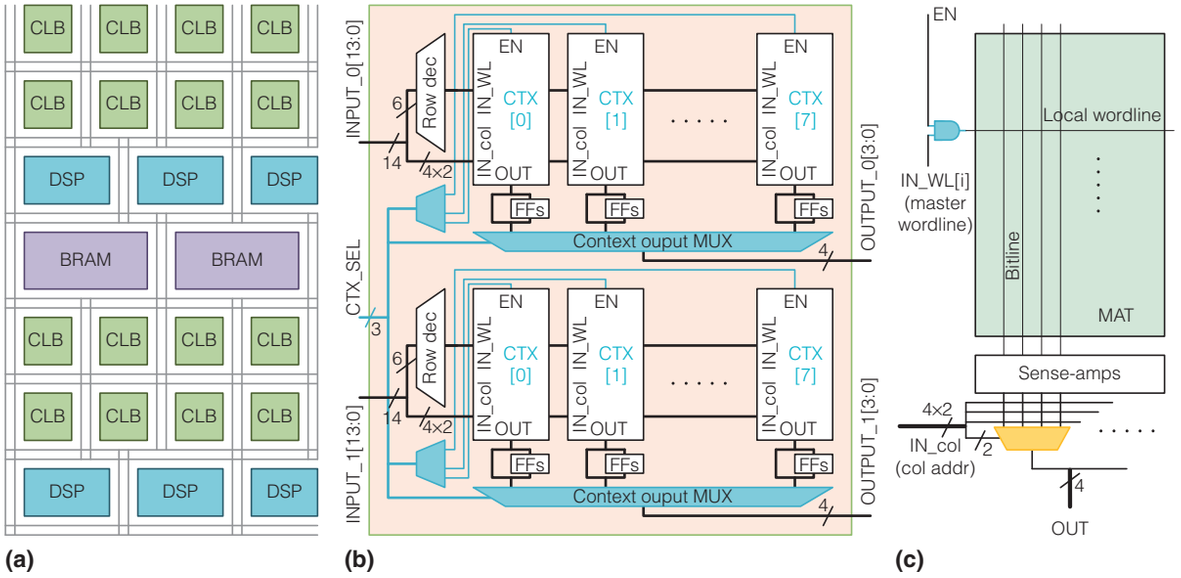

Figure 1. The DRAF architecture. (a) The block layout of DRAF, similar to an FPGA. Block sizes and numbers can vary across devices. (b) The configurable logic block (CLB) in DRAF. As a typical example, a CLB contains two DRAM-based lookup tables and associated flip-flops (FFs) organized into eight contexts. Each lookup table has an 8-bit (6 bits for row and 2 bits for column) input and a 4-bit output. (c) Detailed view of one context in a DRAF lookup table with the context enable gate and specialized column logic.

split into two parts and connected to the row and column address ports of the DRAM array, respectively. To support multicontext, each lookup table is divided into four to eight contexts, leveraging the hierarchical structure in modern DRAM chips, in which the array is divided into DRAM MATs. Each context in DRAF is a modified DRAM MAT (see Figure 1c). The decoded row address will activate a single local wordline, which connects the cells in that row to the bitlines. The data are then transferred to the sense-amplifiers and augmented to full-swing signals.

The typical MAT width and height in commodity DRAM devices are 512 to 1,024 cells. This implies a 9-bit-input, 512-bit-output lookup table, whereas a typical FPGA lookup table has a 6-bit input and 1-bit output. To bring the DRAF lookup table granularity close to the needs of real-world applications to increase the logic utilization, we make each MAT narrower, reducing its width to 8 to 16 bits. This offers a good tradeoff between utilization and density. The aggregated row size of all contexts is still in the order of hundreds of bits, sufficiently amortizing the area overheads of the shared peripheral logic (such as the row decoder).

To further increase the logic utilization and flexibility, we apply a specialized column logic to allow for each output bit to be independently selected from the corresponding set of bitlines. As Figure 1c shows, rather than sharing the same column address for all bits as in conventional DRAM, we organize the 16 bitlines into four groups, and provide each group a separate set of 2 bits to select one output from the 4 bits. This additional level of multiplexing further reduces the output width to 4 bits, while allowing each bit to have partially different input bits, increasing the flexibility of the lookup table functionality.

Multicontext Support

DRAF seamlessly supports multicontext operations by storing each design configuration in one MAT and allowing for single-MAT access. Effectively, each MAT forms one context across all lookup tables. The multiple contexts in one device can be used for different independent logic designs, each of which accelerates one application running

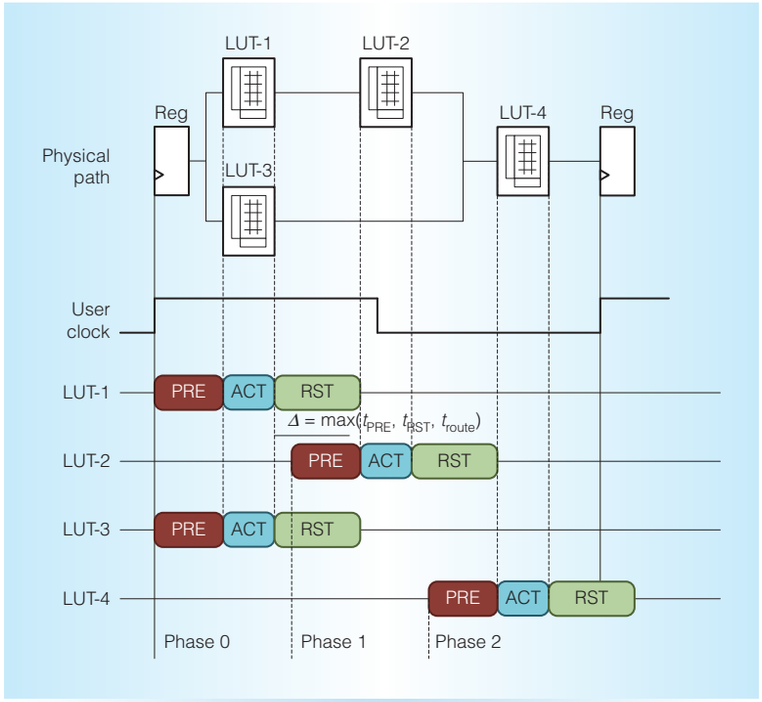

Figure 2. The timing diagram and critical path for a chain of four DRAF lookup tables. Each clock cycle is decomposed into three phases. LUT-1 and LUT-3 are in phase 0, LUT-2 is in phase 1, and LUT-4 is in phase 2. Δ represents the three-way overlapping of restoration, precharging, and routing delays.

on the shared system. Alternatively, we can split a single large and complicated design, such as websearch in Microsoft Catapult,[3] and map each part to one context in order to fit the entire design on a single device instead of using a multi-FPGA network.

We leverage the hierarchical wordline structure in DRAM to decouple the accesses to each MAT by adding an enable AND gate to each local wordline driver,[6] as shown in Figure 1c. This lets us access only a single MAT corresponding to the current selected context, while disabling the other MATs. A context output multiplexer selects the enabled context for the lookup table output port.

The multicontext support in DRAF is particularly efficient. First, the area overhead is negligible, because the peripheral logic (for example, the row decoder) is shared between contexts. Second, the idle contexts (MATs) are not accessed, introducing little dynamic power overhead, and they can be further power-gated to reduce static power. Third, because the design configurations are stored in-place in each lookup table, the context switch is instant by simply updating a context index counter (`CTX_SEL` in Figure 1b) and the new context is ready to use in the next cycle.

Timing Management and Optimization

DRAM access is destructive. Therefore, modern DRAM array organization introduces a two-step access protocol. First, an entire DRAM row is activated and copied into the sense-amplifiers according to the row address (activation); next, a subset of the sense-amplifiers are read or written on the basis of the column address. Because the original charge of the cells in the DRAM row is destroyed after the activation, the cells must be recharged or discharged to restore to the original values (restoration).[7] Finally, we must precharge the bitlines and sense-amplifiers to prepare for the next activation (precharging). The explicit activation, restoration, and precharging create two major challenges for using DRAM in a reconfigurable fabric.

First, when multiple DRAM-based lookup tables are chained together for a large logic function, we must enforce a specific order for each lookup table access and the corresponding timing constraints, to avoid loss of configuration data. DRAF uses a phase-based timing solution. We divide the accelerator design (user) cycle into multiple phases and assign a specific phase for each lookup table in the design (see Figure 2). By requiring the phase of a lookup table to be greater than the phases of all lookup tables producing its input signals, we can guarantee the correct access order. We also delay the precharge operation into the next user cycle, ensuring that the lookup table output is valid across different phases (for example, from LUT-2 and LUT-3 to LUT-4 in Figure 2). The phase assignment can be implemented by a CAD tool using techniques similar to critical path finding. The phase information is stored in a small local controller per lookup table. There is no need for global coordination at runtime.

Second, the restoration and precharging of DRAM arrays introduce high latency overheads[7] that limit the design frequency to no more than 20 MHz. To hide these overheads, DRAF applies a three-way latency overlapping without violating the timing constraints. As

Figure 2 shows, we overlap the charge restoration of the source lookup table that produces a signal, with the time for precharging the destination lookup table of this signal, and the time for routing this signal between the two lookup tables. Because routing delay is typically the dominant latency component in FPGAs,[8] this critical optimization lets DRAF be only two to three times slower than an FPGA and provides reasonable performance speedup over programmable cores.

DRAM Refresh

DRAM requires periodic refresh due to cell leakage. We refresh all lookup tables in a DRAF chip concurrently using a shared row address counter in each CLB and BRAM block. This is easier for DRAF than for commodity DRAM chips in terms of power consumption, because the arrays are much smaller in DRAF. All utilized contexts are refreshed simultaneously, and unused contexts are skipped. The DRAF device driver coordinates the refresh, by holding on to new requests, ignoring output data, and pausing ongoing operations similar to processor pipeline stalls. The internal states in the DRAF design are held in the flip-flops and are not affected. The pause period is less than 1 μs per 64 ms, which is negligible even for latency-critical applications in datacenters that require millisecond-level tail latency.

Design Flow

The success of a reconfigurable fabric relies heavily on the CAD toolchain support. Because DRAF uses the same primitives (lookup tables, flip-flops, DSPs, and BRAMs) as modern FPGAs, its design flow is similar to that of FPGAs with some mild tuning. First, the tool now needs to pack more logic per lookup table to utilize the larger lookup tables. Second, the primary optimization goal should be latency, because area is not a scarce resource anymore. Third, the tool must enforce all timing requirements, including the phases and DRAM timing constraints. Finally, the tool should take advantage of the multicontext support.

Use of DRAF for Datacenter Accelerators

DRAF trades off some of the potential performance of FPGAs to achieve high logic density, multiple contexts, and low power consumption. These features make DRAF devices appropriate for both mobile and server applications, in which one wants to introduce an FPGA device for acceleration without significantly impacting existing systems' power budget, airflow, and cooling constraints.

In datacenters that host public and private clouds, servers are routinely shared by multiple, diverse applications to increase utilization. Different applications and different portions of each application (for example, RPC communication versus security versus main algorithm) require different accelerators. The long reconfiguration latency of conventional FPGAs leads to nonnegligible application downtime,[3] decreasing the system availability and making it expensive to share the acceleration fabrics.

In contrast, DRAF provides a shared fabric that supports multiple accelerators by using different contexts, which can be viewed as multiple independent FPGA instances that need to be used in a time-multiplexed fashion. The high logic density ensures that each individual context has sufficient capacity for the different accelerator designs. The instantaneous context switch ensures that the desired accelerator becomes immediately available to use when needed, with negligible overhead in energy and no application downtime. Being able to share the acceleration fabric can greatly reduce the overall system cost while still enjoying the benefits of special-purpose acceleration.

Evaluation

We evaluate DRAF as a reconfigurable fabric for datacenter applications using a wide set of accelerator designs for representative computation kernels commonly used in large-scale production services, including both latency-critical online services and batch data analytics. We use seven-input, two-output, eight-context lookup tables in DRAF, because they achieve a good tradeoff between efficiency and logic utilization and flexibility. We compare DRAF to an FPGA similar to a Xilinx Virtex-6 device and a programmable processor (Intel Xeon E5-2630 at 2.3 GHz). For a fair comparison, the accelerator designs are synthesized using the same open-source CAD tools for both the conventional FPGA and DRAF. The DRAF

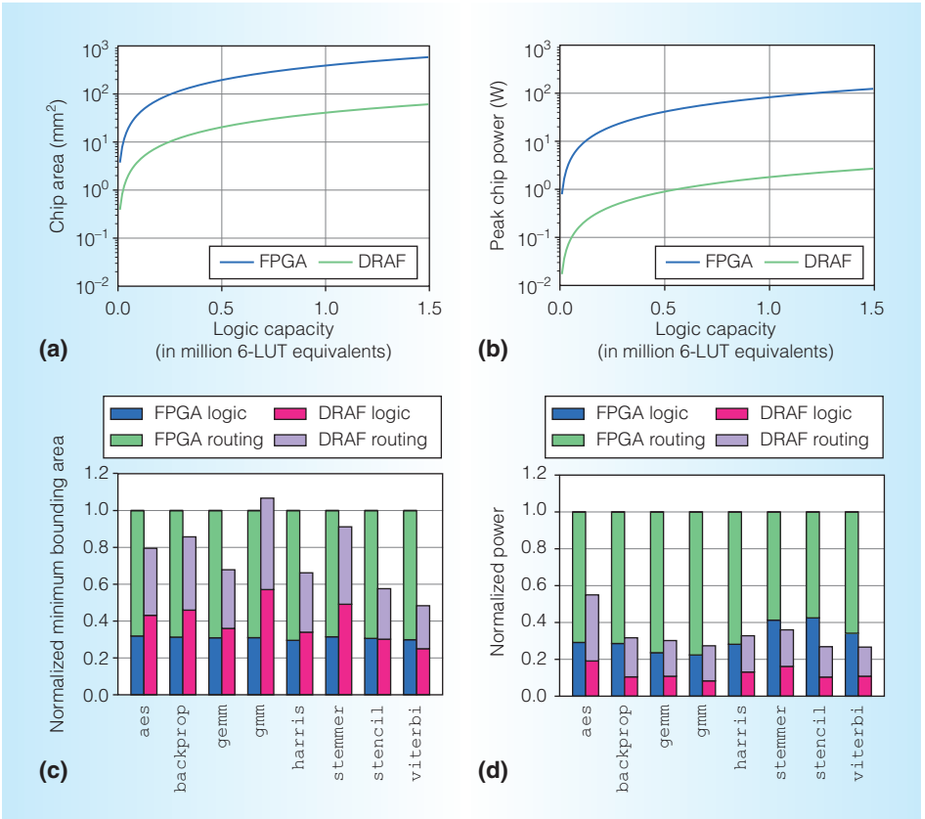

Figure 3. Area and power comparison between DRAF and a conventional FPGA. (a, b) Device-level comparison. (c, d) Comparison after real application mapping.

results are conservative compared to the programmable core baselines, because highly optimized commercial tools are likely to generate more efficient mappings of accelerator designs on the DRAF fabric. Our full paper contains a complete description of our methodology.[9]

Area and Power

Figures 3a and 3b compare the area and peak power consumption of DRAF and FPGA devices with different logic capacities measured in 6-bit-input lookup table equivalents for 45-nm technology. For a fixed logic capacity, an eight-context DRAF provides more than 10 times area improvement and roughly 50 times power consumption reduction. If we target a cost-effective device size of 75 mm^2, an FPGA can pack roughly 200,000 lookup tables, whereas DRAF can have more than 1.5 million lookup tables, a logic capacity comparable to that of the state-of-the-art Virtex-UltraScale+ FPGAs that use a much more recent 16-nm technology. The power consumption advantage is also remarkable. Although the FPGA power can easily exceed 10 W, DRAF consumes only about 1 to 2 W.

Figures 3c and 3d further compare DRAF to FPGA using real accelerator designs. We map each accelerator to one of the eight available contexts in DRAF. The other unused contexts still contribute to the area, consume leakage power, and introduce a slight access latency penalty in the DRAF lookup tables. On average, each accelerator design occupies 19 percent less area on DRAF than on the FPGA, roughly matching the 10-times area advantage if we consider the seven additional contexts available within the area occupied in DRAF. DRAF's area advantage stems primarily from using lookup tables with wider inputs and outputs; these lookup tables can realize larger functions and also reduce

pressure on the configurable interconnect. The `gmm` design uses more area in DRAF than the FPGA, because it requires exponential and logarithmic functions that are not currently supported in our DSP blocks.

Regarding power, the FPGA power consumption is dominated by the routing fabric, especially for larger designs. DRAF provides a 3.2 times power improvement on average, resulting from both the more efficient DRAM-based lookup tables and the savings on routing due to denser packing.

Performance

Finally, we compare the performance of accelerator designs mapped onto DRAF and FPGA devices to that of optimized software running on general-purpose programmable cores. For the programmable cores, we optimistically assume ideal linear scaling to four cores, owing to the abundant request-level parallelism in datacenter services. The chip size for FPGA and DRAF is fixed at 75 mm^2.

Figure 4 shows that both FPGA and DRAF outperform the single-core baseline, on average by 37 and 13 times, respectively. When compared to four cores with ideal speedup, DRAF still exhibits significant speedup of 3.4 times while consuming just 0.63 W, compared to 7 to 10 W of a single core in Xeon-class processors. Overall, these results establish DRAF as an attractive and flexible acceleration fabric for cost (area) and power constrained environments.

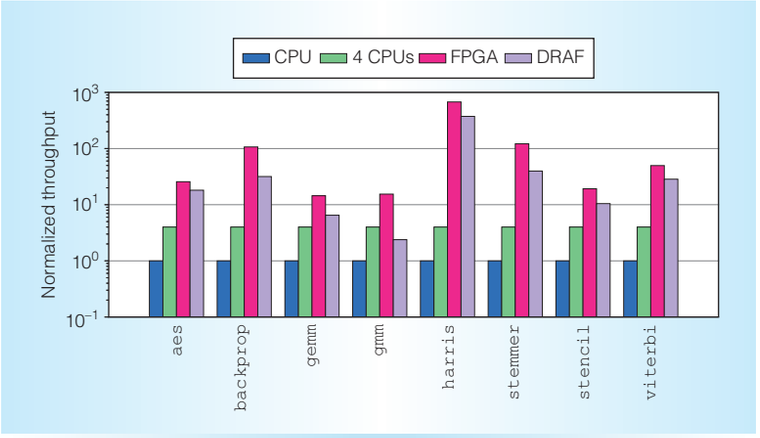

Figure 4. Performance comparison between single-core, multicore, FPGA, and DRAF using representative datacenter application kernels. Assume ideal scaling from single-core to multicore platform.

DRAF is the first complete design to use dense, commodity DRAM technology to implement a reconfigurable fabric with significant logic density and power improvements over conventional FPGAs. DRAF provides a low-cost solution for multicontext acceleration fabrics, which are expected to become widely used in future multitenant cloud and mobile systems. Looking forward, it is important to tune CAD tools and runtime management systems to efficiently map accelerator designs on DRAF, taking full advantage of its high-density and multicontext features.

The techniques that allow DRAF to turn dense storage technology to cost-effective reconfigurable fabrics are also applicable to memory technologies beyond DRAM. The upcoming dense nonvolatile memory technologies, such as spin-transfer torque RAM, exhibit good density scaling and have better static power characteristics compared to DRAM. An exciting research direction is to extend DRAF to exploit the advantages and address the shortcomings of new memory technologies in order to produce acceleration fabrics with low area and power cost. MICRO

References

1. M. Horowitz, "Computing's Energy Problem (and What We Can Do About it)," *Proc. IEEE Int'l Solid-State Circuits Conf.* (ISSCC), 2014, pp. 10–14.
2. R. Tessier, K. Pocek, and A. DeHon, "Reconfigurable Computing Architectures," *Proc. IEEE*, vol. 103, no. 3, 2015, pp. 332–354.
3. A. Putnam et al., "A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services," *Proc. 41st Ann. Int'l Symp. Computer Architecture* (ISCA), 2014, pp. 13–24.
4. S. Trimberger et al., "A Time-Multiplexed FPGA," Proc. 5th IEEE Symp. FPGA-Based Custom Computing Machines (FCCM), 1997, p. 22.
5. B. Van Essen et al., "Static versus Scheduled Interconnect in Coarse-Grained Reconfigurable Arrays," *Proc. Int'l Conf. Field Programmable Logic and Applications* (FPL), 2009, pp. 268–275.

6. A.N. Udipi et al., "Rethinking DRAM Design and Organization for Energy-Constrained Multi-cores," *Proc. 37th Ann. Int'l Symp. Computer Architecture* (ISCA), 2010, pp. 175–186.
7. Y.H. Son et al., "Reducing Memory Access Latency with Asymmetric DRAM Bank Organizations," *Proc. 40th Ann. Int'l Symp. Computer Architecture* (ISCA), 2013, pp. 380–391.
8. V. Betz et al., *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, 1999.
9. M. Gao et al., "DRAF: A Low-Power DRAM-based Reconfigurable Acceleration Fabric," *Proc. ACM/IEEE 43rd Ann. Int'l Symp. Computer Architecture* (ISCA), 2016, pp. 506–518.

Mingyu Gao is a PhD student in the Department of Electrical Engineering at Stanford University. His research interests include energy-efficient computing and memory systems, specifically on practical and efficient near-data processing for data-intensive analytics applications, high-density and low-power reconfigurable architectures for datacenter services, and scalable accelerators for large-scale neural networks. Gao received an MS in electrical engineering from Stanford University. He is a student member of IEEE. Contact him at mgao12@stanford.edu.

Christina Delimitrou is an assistant professor in the Departments of Electrical and Computer Engineering and Computer Science at Cornell University, where she works on computer architecture and distributed systems. Her research interests include resource-efficient datacenters, scheduling and resource management with quality-of-service guarantees, disaggregated cloud architectures, and cloud security. Delimitrou received a PhD in electrical engineering from Stanford University. She is a member of IEEE and ACM. Contact her at delimitrou@cornell.edu.

Dimin Niu is a senior memory architect in the Memory Solutions Lab in the US R&D center at Samsung Semiconductor. His research interests include computer architecture, emerging nonvolatile memory technologies, and processing-near/in-memory architecture. Niu received a PhD in computer science and engineering from Pennsylvania State University. Contact him at dimin.niu@ssi.samsung.com.

Krishna T. Malladi is a staff architect in the Memory Solutions Lab in the US R&D center at Samsung Semiconductor. His research interests include next-generation memory and storage systems for datacenter platforms. Malladi received a PhD in electrical engineering from Stanford University. Contact him at k.tej@ssi.samsung.com.

Hongzhong Zheng is a senior manager in the Memory Solutions Lab in the US R&D center at Samsung Semiconductor. His research interests include novel memory system architecture with DRAM and emerging memory technologies, processing in memory architecture for machine learning applications, computer architecture and system performance modeling, and energy-efficient computing system designs. Zheng received a PhD in electrical and computer engineering from the University of Illinois at Chicago. He is a member of IEEE and ACM. Contact him at hz.zheng@ssi.samsung.com.

Bob Brennan is the senior vice president of the Memory Solutions Lab in the US R&D center at Samsung Semiconductor. He has led numerous research projects on memory system architecture, SoC architecture, CPU validation, and low-power system design. Brennan received an MS in electrical engineering from the University of Virginia. Contact him at bob.brennan@ssi.samsung.com.

Christos Kozyrakis is an associate professor in the Departments of Electrical Engineering and Computer Science at Stanford University, where he investigates hardware architectures, system software, and programming models for systems ranging from cell phones to warehouse-scale datacenters. His research interests include resource-efficient cloud computing, energy-efficient computing and memory systems for emerging workloads, and scalable operating systems. Kozyrakis has a PhD in computer science from the University of California, Berkeley. He is a fellow of IEEE and ACM. Contact him at kozyraki@stanford.edu.